Remarks

Applicant respectfully requests that this Amendment After Final Action be admitted under 37 C.F.R. § 1.116.

Applicant submits that this Amendment presents claims in better form for consideration on appeal. Furthermore, applicant believes that consideration of this Amendment could lead to favorable action that would remove one or more issues for appeal.

No claims have been amended. No claims have been canceled. Therefore, claims 1-26 are now presented for examination.

In the Final Office Action, claims 1-26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Archambault (U.S. Patent No. 6,173,444) in view of Blainey (U.S. Patent No. 6,045,585) in view of "Restricted Pointers are Coming" by Robison. Applicant submits that the present claims are patentable over any combination of Archambault, Blainey, and Robison.

Archambault discloses a method that reduces the size of alias sets associated with program pointers. First, intraprocedural information about pointer variables referenced in each function of the program is gathered and saved in a data structure called a pointer alias graph. Next, the pointer alias graphs from all the compilation units for the program are combined to form a universal pointer alias graph and then transitive closure is performed on the universal pointer alias graph to produce a reduced graph containing the list of objects that each pointer variable can point to. Finally, all the files are re-compiled using the universal pointer alias graph as input, resolving all occurrences where pointer variables are de-referenced. See Archambault at Abstract.

Standard data flow gathering techniques are used to develop a pointer alias graph. The nodes in the graph represent either a definition of a pointer variable or a use of a pointer variable, and each node has an associated alias set. The initial alias sets for the nodes of the graph are defined as follows: the initial alias set for definition nodes is the right hand side of the pointer variable assignment operation, and the initial alias set for use nodes is the value of the object at that execution point (the r-val). Location information, the basic block number (relative to the flow graph) and position within the basic block, is saved for each node. In order to provide a complete representation of pointer use in the function for the interprocedural analysis, global unique names, called pseudo pointer variables, are assigned to each formal argument, function return value and global (or file scope) variable. Corresponding nodes and alias sets are created on the pointer alias graph (col. 5, ll. 4-17).

However, Archambault does not disclose or suggest a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer. In fact, the Examiner, in the Final Office Action admits that "Archambault does not expressly disclose the limitation wherein at least one of the number of pointers is a restricted pointer." See Final Office Action at page 4, paragraph 1. Instead, the Examiner cites Blainey as including such a feature.

Blainey discloses a system and method for determining alias information at a inter-compilation unit level of a compilation process. The method includes the steps of determining anti-alias sets from the alias information provided by a first stage of the compilation process, calculating pessimistic inter-compilation unit alias sets and refining these sets, after transitive closure as appropriate, with the anti-alias sets. See Blainey at

Abstract. Nonetheless, Blainey does not disclose or suggest a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer.

Robison discloses the uses of restricted pointers in C and C++. See Robison at paragraph 1. However, Robison does not discuss a method for optimizing C and C++ code that include restricted pointers. Therefore, Robison does not disclose or suggest a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer.

Claim 1 of the present application recites:

> A method comprising:
> receiving a code segment having a plurality of instructions, the code segment having an outer scope and a number of inner scopes, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer; and
> determining, within one of the number of inner scopes, whether at least two pointers of the number of pointers are aliases.

As discussed above, Archambault, Blainey and Robison do not disclose or suggest a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer. Even though both Archambault and Blainey disclose methods for compiler optimization, they do not disclose optimization of code with restricted pointers. Thus, neither disclose or suggest a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer. The Examiner asserts that

> Blainey discloses examples of such language features and programmer assertions that relate to pointers and memory access (see column 2,

> lines 40-46). Another comparable language feature
> is, for example, restricted pointers.

See Final Office Action at page 2, paragraph 4. Applicant fails to understand how

obtaining aliasing rules based upon language rules or assertions suggest a code segment

having a plurality of instructions including a number of pointers wherein at least one of

the pointers is a restricted pointer. The present application describes a restricted pointer

as a special type of pointer. There is no disclosure in Blainey that the language rules or

assertions suggest a special type of pointer. On the contrary, Blainey discloses rules that

define how a pointer may be used, not a type of pointer like a restricted pointer. As

discussed above, Robison simply discloses the uses of restricted pointers in C and C++,

there would be no motivation to combine with Archambault and Blainey, since neither

Archambault nor Blainey disclose the use of restricted pointers.

Therefore, claim 1 is also patentable over Archambault in view of Blainey further

in view of Robison. Claims 2-7 depend from claim 1 and include additional limitations.

Thus, claims 2-7 are also patentable over Archambault in view of Blainey further in view

of Robison.

Claim 8 recites:
> A method comprising:
> <u>receiving a code segment having a plurality of
> instructions, wherein the plurality of instructions
> comprise a number of pointers, wherein at least one
> of the number of pointers is a restricted pointer,</u> and
> wherein the at least one restricted pointer is in-
> scope or out-of-scope; and
> determining whether at least two pointers of the
> number of pointers are aliases when each pointer of
> the at least two pointers is out-of-scope relative to
> the other pointers of the at least two pointers.

Accordingly, for the reasons described above with respect to claim 1, claim 8 is also patentable over Archambault in view of Blainey further in view of Robison. Since claims 9-12 depend from claim 8 and include additional limitations, claims 9-12 are also patentable over Archambault in view of Blainey further in view of Robison.

Claim 13 recites:

> A system comprising:
> a memory unit to include <u>a code segment having a plurality of instructions, the code segment having an outer scope and a number of inner scopes, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer</u>; and
> a compiler unit coupled to the memory, the compiler unit to determine within one of the number of inner scopes, whether at least two pointers of the number of pointers are aliases.

Accordingly, for the reasons described above with respect to claim 1, claim 13 is also patentable over Archambault in view of Blainey further in view of Robison. Since claims 14-16 depend from claim 13 and include additional limitations, claims 14-16 are also patentable over Archambault in view of Blainey further in view of Robison.

Claim 17 recites:

> A machine-readable medium that provides instructions, which when executed by a machine, cause said machine to perform operations comprising:
> <u>receiving a code segment having a plurality of instructions, the code segment having an outer scope and a number of inner scopes, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer</u>; and
> determining, within one of the number of inner scopes, whether at least two pointers of the number of pointers are aliases.

Accordingly, for the reasons described above with respect to claim 1, claim 17 is also patentable over Archambault in view of Blainey further in view of Robison. Since claims 18-21 depend from claim 17 and include additional limitations, claims 18-21 are also patentable over Archambault in view of Blainey further in view of Robison.

Claim 22 recites:

> A machine-readable medium that provides instructions, which when executed by a machine, cause said machine to perform operations comprising:
>
> receiving a code segment having a plurality of instructions, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer, and wherein the at least one restricted pointer is in-scope or out-of-scope; and
>
> determining whether at least two pointers of the number of pointers are aliases when each pointer of the at least two pointers is out-of-scope relative to other pointers of the at least two pointers.

Accordingly, for the reasons described above with respect to claim 1, claim 22 is also patentable over Archambault in view of Blainey further in view of Robison. Since claims 23-26 depend from claim 22 and include additional limitations, claims 23-26 are also patentable over Archambault in view of Blainey further in view of Robison.

Applicant respectfully submits that the rejections have been overcome, and that the claims are in condition for allowance. Accordingly, applicant respectfully requests the rejections be withdrawn and the claims be allowed.

The Examiner is requested to call the undersigned at (303) 740-1980 if there remains any issue with allowance of the case.

Please charge any shortage to our Deposit Account No. 02-2666.

                                    Respectfully submitted,
                                    BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: August 12, 2004   _____
                                    Mark L. Watson
                                    Reg. No. 46,322
12400 Wilshire Boulevard
7th Floor
Los Angeles, California 90025-1026
(303) 740-1980